Heritage Science

RESEARCH ARTICLE

Open Access

Semantic segmentation and photogrammetry of crowdsourced images to monitor historic facades

Ziwen Liu[1*], Rosie Brigham[1], Emily Rosemary Long[1], Lyn Wilson[2], Adam Frost[2], Scott Allan Orr[1] and Josep Grau-Bové[1]

Abstract

Crowdsourced images hold information could potentially be used to remotely monitor heritage sites, and reduce human and capital resources devoted to on-site inspections. This article proposes a combination of semantic image segmentation and photogrammetry to monitor changes in built heritage sites. In particular, this article focuses on segmenting potentially damaging plants from the surrounding stone masonry and other image elements. The method compares different backend models and two model architectures: (i) a one-stage model that segments seven classes within the image, and (ii) a two-stage model that uses the results from the first stage to refine a binary segmentation for the plant class. The final selected model can achieve an overall IoU of 66.9% for seven classes (54.6% for one-stage plant, 56.2% for two-stage plant). Further, the segmentation output is combined with photogrammetry to build a 3D segmented model to measure the area of biological growth. Lastly, the main findings from this paper are: (i) With the help of transfer learning and proper choice of model architecture, image segmentation can be easily applied to analyze crowdsourcing data. (ii) Photogrammetry can be combined with image segmentation to alleviate image distortions for monitoring purpose. (iii) Beyond the measurement of plant area, this method has the potential to be easily transferred into other tasks, such as monitoring cracks and erosion, or as a masking tool in the photogrammetry workflow.

Keywords: Cultural heritage, Crowdsourced image processing, Deep learning, Structure from Motion, Remote sensing, Dilated convolution

Introduction

Preservation of the authenticity and physical integrity of heritage sites requires regular monitoring and maintenance through on-site inspections. Although inspections can protect sites from a variety of natural and anthropogenic threats, they require intensive human and material resources, especially for remote sites. For example, Historic Environment Scotland (HES) manages approximately 300 properties, with some of the sites being extremely remote [1]. In addition to on-site inspections, managers of those sites deploy remote equipment, or ask citizen scientists and visitors to collect data to assist with preservation works. Given that HES properties attracted 5 million visitors in 2018 [2], the photographs and videos from visitors could create a beneficial crowdsourced visual resource. In particular, this data can capture common forms of deterioration in cultural heritage, such as erosion, cracks, plant growth and other physical damage, that may accumulate over time to more severe damages, reducing the aesthetic value of the heritage sites [3, 4].

Crowdsourcing has been widely used in the past to assist heritage preservation works [5]. Examples of dedicated crowdsourcing projects that successfully helped

*Correspondence: z.liu.19@ucl.ac.uk
[1] University College London, Central House, 14 Upper Woburn Place, London WC1H 0NN, UK
Full list of author information is available at the end of the article

preservation and reconstruction works include [6, 7] and [8]. However, the extraction and analysis of crowdsourced imagery data becomes an issue for cultural heritage practitioners as a consequence of two features brought by crowdsourcing. First, due to the diversity of camera angles of crowdsourced images, properties (e.g. areas of objects) cannot be directly compared between distinctive images. Second, as crowdsourced images were taken by multifarious photographic equipment under different lighting conditions, the sizes and qualities of these images are various. As a result, objects within the images are difficult to be recognized automatically. Thus, this article tests a combination of computer vision and photogrammetry to measure potentially damaging plant growth from crowdsourced images of built heritage sites.

Within computer vision, convolutional neural networks (CNNs) are an effective tool for image analysis because they can learn high-level features of images [9]. Three types of CNNs have been applied to built heritage datasets: classification, object detection, and semantic segmentation. In particular, these CNNs have proved helpful for the documentation and damage detection in masonry walls. To identify spalling, cracks, and efflorescence in bricks at the Palace Museum in Beijing, researchers tested image classification CNNs with a sliding window-based approach [10]. This method was later improved by using an object detection algorithm that can identify and classify bricks regardless of their size or placement within the wall [11]. Multi scale image segmentation has been implemented to identify polygonal stones in castellated walls which were then stored in a stone management database [12, 13]. Additionally, semantic segmentation can detect objects that occlude brick walls, so a predicted pattern for the brick and mortar can be referenced in case the wall is damaged [14].

The usefulness of CNNs for built heritage goes beyond identifying weathering in historic stones. Image classification CNNs have identified mould and deterioration on the interior of buildings [15]. Object detection algorithms have detected components of ventilation systems in historic libraries [16], and missing decorative tiles from ancient roofs [17]. Finally, semantic segmentation has been used to identify and quantify the damage to yellow-glazed roof tiles [18]. CNNs are incredibly versatile in their applications to built heritage, but there is one type of damage that is underrepresented in the literature.

Plant growth can be particularly harmful, both chemically and mechanically, to stones in historic structures [19]. Identifying and measuring the area of new plant growth from crowdsourced images could prevent further damage. Past solutions to plant measurement relied on manually selecting pixels with image processing software, such as ImageJ. Some newer software specific to leaf area measurement, such as Easy Leaf Area [20] and Leaf-IT [21], incorporated traditional automatic image segmentation techniques [22, 23]. However, these methods are prone to misclassifications for more complex images, like those of cultural heritage sites that have similar colors and unclear margins between the background and target objects. CNNs have successfully identified images with plants growing between historic stones [19], and agricultural research has shown that semantic segmentation can detect plants within complex scenes [24]. Thus, this article will propose a semantic segmentation approach to separate plants from other elements within crowdsourced images of built heritage.

There is no 'one-size-fits-all' solution for monitoring and documenting a heritage site, but photogrammetry has proved to be a popular method [25–27]. Some sources argue that terrestrial laser scanning (TLS) is more accurate than photogrammetry [28]. However, photogrammetry lends itself well to damage detection because it offers higher resolution textures than TLS [29]. While photogrammetric surveys create effective 3D models for damage detection in heritage buildings, they are time-consuming if the damage has to be manually labelled [29]. Previous automated segmentation methods are ineffective on larger and more complex datasets [30]. Further attempts of automating change detection between long-term photogrammetric surveys led to outputs that are hard to interpret [31]. The most promising new way to segment photogrammetric models for built heritage is to implement a semantic segmentation CNN.

State-of-the-art papers have investigated the use of semantic segmentation algorithms to automate the process of segmenting point clouds by architectural elements for 3D heritage documentation [32–36]. However, these methods were tested on large-scale photogrammetry surveys or laser scans, which may be impractical or prohibitively expensive for some heritage sites. By mobilising visitors to submit crowdsourced images, photogrammetry and semantic segmentation have the potential to quickly produce small-scale 3D models that provide updates on the site.

This study, for the first time in the authors' knowledge, applies image segmentation algorithms to photogrametric models of to extract useful information from crowdsourced photographs to support remote monitoring activities at cultural heritage site.

Study site and data

The crowdsourced images used in this article are from Monument Monitor [37], a collaborative research project between HES and the Institute of Sustainable Heritage at University College London. The project aims to facilitate conservation and monitoring efforts at

heritage sites by extracting useful information from visitor photographs. Through signage around the heritage site, visitors are asked to submit their photographs via email or social media platforms such as Facebook and Twitter. Approximately 20 properties across Scotland are under the supervision of this project [37]. This paper focuses on images of Bothwell Castle, a large medieval castle located in South Lanarkshire, Scotland. The site dates back to the thirteenth century and was damaged by sieges during the Wars of Independence [38]. The main scene of these images is an inner corner of the Bothwell castle with a small locked gate at the south-east corner of the site, as shown in the first row of Fig. 1.

In total, there are 113 photographs of Bothwell Castle in the dataset. These photographs were taken between January 2019 and March 2020, and they are stored in JPEG and PNG format. The size and resolution of the images vary greatly across the dataset since they were submitted by different visitors with a range of photographic equipment. The dataset was split into a training set of 93 images and a test set of 20 images. These images were manually labelled with Labelbox [39]. There are seven classes in total, including window (108/113), sky (110/113), plant (101/113), masonry (113/113), hole (83/113), gate (43/113) and signboard (36/113). Some examples of labelled images are presented in the second row of Fig. 1.

Materials and methods

Image segmentation algorithm

Semantic image segmentation

The creation of the deep convolutional neural network AlexNet [40] drastically improved the performance of computer vision algorithms for classification. The image classification error rate of the ImageNet competition dramatically reduced from 2011 to 2017 [41]. Then CNNs were adapted to perform computer vision tasks beyond overall image classification, including semantic image segmentation. As shown in Fig. 1, semantic segmentation involves detecting and partitioning an image into different segments based on their class. According to a literature review [42], recent popular image segmentation models are usually made up of two parts, namely a backend and a classifier. The backend is a deep CNN which is responsible for extracting the features of an image to form a feature map through convolutional and pooling operations. The classifier, with a relatively small neural network architecture compared to the backend, is responsible for making predictions based on the feature map from the backend. Finally, the prediction is bilinearly interpolated to size of the original image, and they are combined to form a segmented image. This process is summarized by Fig. 2.

An image segmentation model classifies each pixel based on the predicted probabilities of a pixel belonging to each possible class. This is achieved by finding optimal weights of the neural network by minimizing the loss

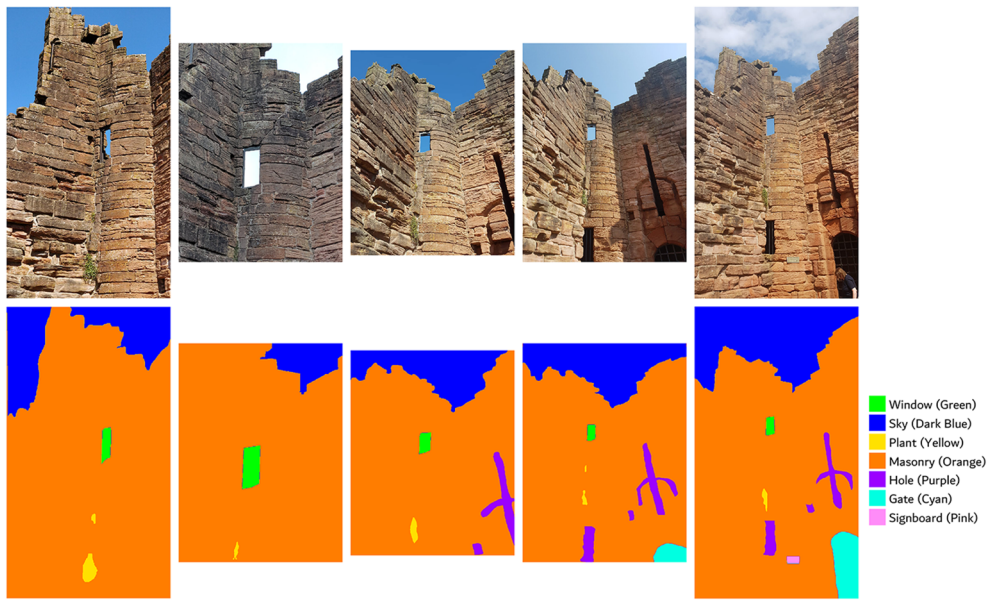

Fig. 1 A group of representative examples of images from the dataset. The original crowdsourced photographs are in the top row, and their corresponding segmentation labels are shown

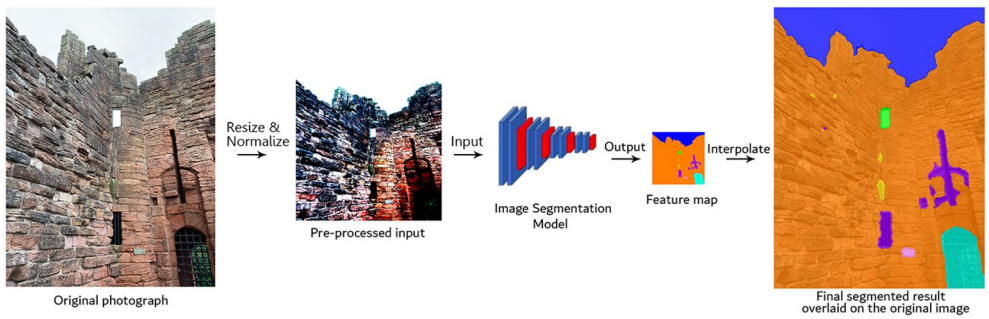

Fig. 2 The summarized process of image segmentation

function. The loss measures the difference between the ground truth label y and predicted one $\hat{y}$ from the training phase. A weighted cross-entropy loss function of Eq. (1) is used for image segmentation in this case, where $i = 1, \ldots, C$ is the class of the pixel. The weight for class W_i is calculated based on Eq. (2), where f_i is the frequency of the pixels of class i in the training set. This is to avoid the model to favor small objects too heavily, or ignore small objects and focus on large objects in prediction.

$$Loss(y, \hat{y}) = -\sum_{i=1}^{C} W_i \cdot y_i \cdot log(\hat{y}_i) \tag{1}$$

$$W_i = \frac{log(f_i)}{\sum_{j=1}^{C} log(f_j)} \tag{2}$$

Transfer learning is a technique which allows a deep-learning model to be adjusted and applied to another, usually smaller dataset [42]. The backbones of segmentation models can first be trained with a computer vision competition dataset, usually exceeding 100 k labelled training examples, such as COCO [43] or the Open Images dataset [44]. These large datasets contain images of complex yet common scenes from daily life. When a model has already been trained to extract features from universal images, transfer learning then saves tremendous computational cost and time. As lack of high-quality cultural heritage dataset limits the application of machine learning methods in heritage studies [45], transfer learning becomes an important novel technique to bring general knowledge learnt from other areas to solve data-sparsity problem in the heritage domain. This paper innovatively combines transfer learning with photogrammetry to highlight the potentiality of transfer learning technique in improving heritage monitoring work when combined with existing methods.

There are several image segmentation algorithms that are based on deep convolutional neural networks and transfer learning techniques, including DeepLab family models [46], FCN [47], U-Net [48] and Mask R-CNN [49]. Considering the exceptional performance in similar tasks and relatively easy implementation and preparation works compared to other algorithms, DeepLab [46] family models are adopted in this task.

DeepLab family and DeeplabV3

DeepLab family is a representative of dilated convolutional models. As stated in the papers of DeepLab family models [46, 50], compared to other computer vision tasks, image segmentation faces two main problems. First, the small resolution of the produced feature map due to multi-layer convolutional operations makes it difficult to detect small objects and draw clear boundary in an image, such as plant or cracks with small areas in this case. Second, the varying scales of the same type of objects within different images can influence the performance of image segmentation models. This issue will be prominent for tasks dealing with unstructured imagery data (e.g. crowdsourced photographs).

To address those problems, DeepLab family models propose to use a dilated convolution operation, which adds 'holes' in the convolution kernel to skip some pixels. The rate of dilation is the number of zeros between two consecutive filter values along each spatial dimension as illustrated by Fig. 3. On the basis of dilated convolution, Deeplabv family models apply Atrous Spatial Pyramid Pooling (ASPP) to extract features in different scales and produce a higher resolution feature map with only minimal additional computational cost, which obtained an outstanding performance on dealing with small objects segmentation and varying scales problem. Specifically, using DeeplabV3 as an example, the model uses convolutional layers of a 3×3 kernel with different rates of dilation. After processing the feature map in parallel, it concatenates the outputs from the dilated convolution layers and an output from a global average pooling layer. This

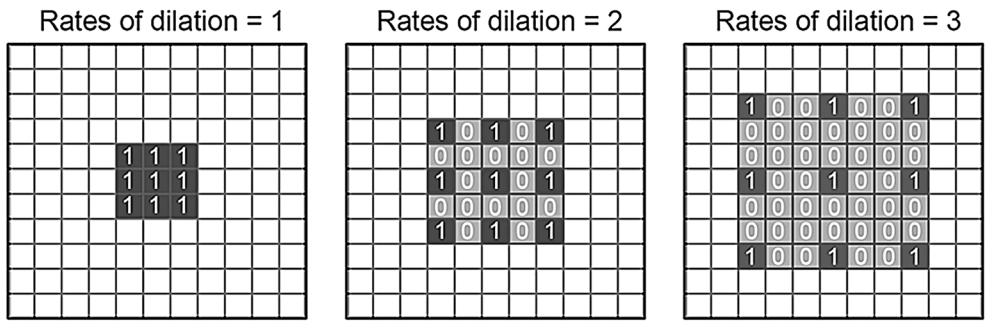

Fig. 3 An illustration of dilated convolution, where a rate of dilation equal to 1 is the original convolution kernel with no dilation

is further processed by a 1 × 1 convolution layer and bilinearly interpolated to form a tensor to be used in the final calculation of loss. This strategy enables the model to not only save the required memory of GPU in computation due to no computation in the dilated convolution kernel, but also gain a larger field-of-views resulted from the expanded convolution kernel and the ability to deal with varying scale objects due to the multi-grid ASPP. In this article, the DeeplabV3 with a backend part of ResNet101, abbreviated as Model1, will serve as the baseline model to be compared with other models.

DeeplabV3+

To further improve performance, DeeplabV3+ [51], the latest version of the DeepLab family of image segmentation models, reconsidered its structure as an encoding-decoding structure (with DeeplabV3 structure as the encoding part) and added a more sophisticated decoding part. The encoding step downsamples the image into a small prediction map. Then the decoding step upsamples the prediction map into the original size of the input image. In the decoding stage, it concatenates the features processed by an 1 × 1 convolution operation from the feature map produced by the backbone model. This is followed by another 3 × 3 convolution layer before interpolating to the original image size, rather than a naïve bilinear interpolation upsampling. This gives the model a better ability of predicting a smoother and precise result given the input image.

Another modification of DeeplabV3+ is that, apart from ResNet, it utilises a modified version of the Xception network [52]. With several changes in the backbone neural network structure, it achieved exceptional image segmentation results. DeeplabV3+ with two distinctive backbone parts, namely Xception and ResNet, was applied to this crowdsourced dataset. They will be referred to as Model2 and Model3 respectively.

Two-stage model design

The previous models partition whole images into distinguishing segments belonging to different classes. Additionally, this paper explores an optional two-stage model that only makes a binary classification to further refine the segmentation results for a specific class. As illustrated by Fig. 4, this second model type classifies pixels into plant and non-plant classes from a crop based on the bounding box of plants from the output of the first segmentation model. This cropped image is given by reverse selection using density-based spatial clustering of applications with noise (DBSCAN) [53], an unsupervised clustering technique, to partition and assign labels to disjoint individuals in the output of the image segmentation model. *eps* in DBSCAN controls the maximum distance between two pixels for one to be considered as in the same region of the other. It is therefore a free parameter for this algorithm that can be adjusted according to different scenarios.

The second model is also a DeeplabV3+ model with ResNet as its backbone part using the same procedure as the first model (except there is no log-transformation of the weights to each class in the loss function). The original image dataset was manually cropped around all the plants. Therefore, the training of this two-stage model does not require additional photographs. This extra step was carried out to refine the prediction result by further clearing the boundary between plants and non-plant objects and provide a more flexible solution to segmenting objects in different scenes by adding the parameter *eps*.

Photogrammetry

Although crowdsourced images can provide insights into the plant growth at heritage sites, it is inaccurate to compare areas of plant growth from photographs taken from very different angles. To address this, photogrammetry can be used to build a comprehensive and stereoscopic

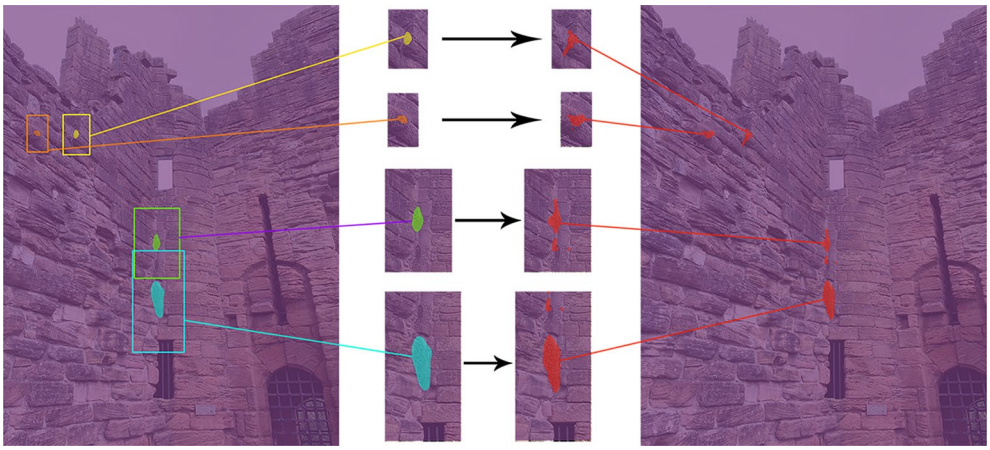

Fig. 4 Illustration of two-stage model. The second model takes the cropped part of the first model's prediction (objects in the same class are separated into different disjoint instances by DBSCAN algorithm) to further refine the segmentation result

view of a heritage site by constructing a 3D model from crowdsourced photographs. The photogrammetry method used in this article is incremental Structure from Motion (SfM) [54]. To reconstruct a complete 3D model from 2D images, the process of photogrammetry can be roughly split into three steps, namely features matching, sparse reconstruction and dense reconstruction. The first step detects common features among the input images using algorithms such as Scale-Invariant Feature Transform (SIFT) [55]. Then, the second step uses bundle adjustment [56] to form estimated camera poses and an optimal sparse 3D model. Finally, the last step reconstructs a dense 3D model with the Patch-Match algorithm [57].

This article combines photogrammetry with image segmentation, as shown in Fig. 5. Firstly, photogrammetry software is used to construct a 3D model of the inner corner of Bothwell Castle from the crowdsourced image dataset. Next, the same image dataset is partitioned into segments locating each class by the image segmentation model. Finally, the segmented images are re-mapped onto the 3D model, given the known camera poses and

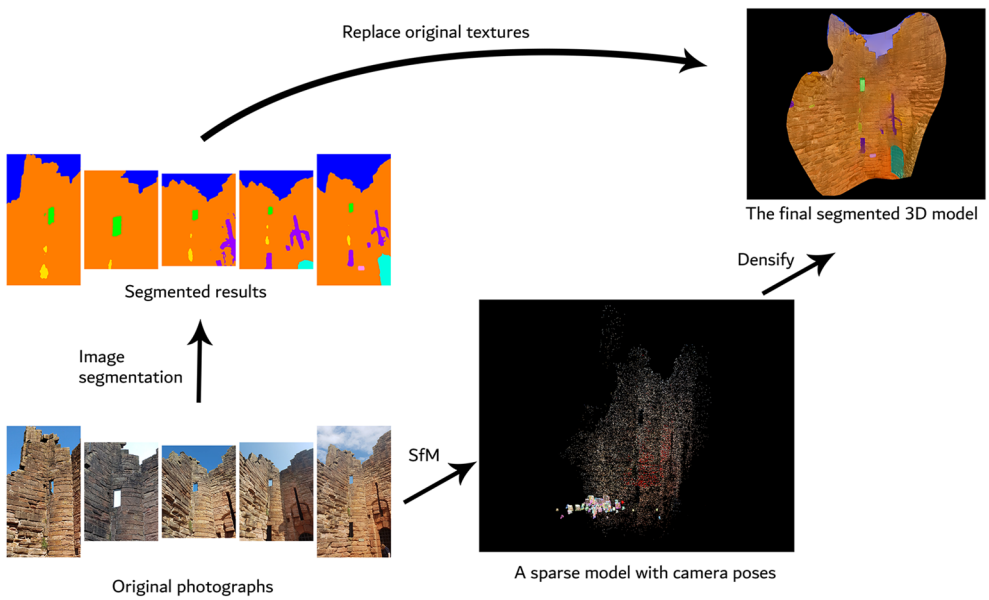

Fig. 5 The process of combining image segmentation and photogrammetry

mapping scheme. This gives a 3D model with textures segmented into different regions. Hence, a property of an object, such as the plant area, can be easily measured from any viewpoint to remove the distortion caused by angle of shooting. Lastly, a flowchart of Fig. 6 summarizes the whole process of the proposed algorithm.

Software and hardware

This paper uses PyTorch as the framework of the image segmentation models, combined with its associate computer vision library Torchvision and an implementation of DeeplabV3+ [58]. Image processing was conducted using the open-source computer vision and photogrammetry tools OpenCV [59], VisualSFM [60] and OpenMVS [61]. As for the hardware, the model is trained on a personal computer with a CPU of Intel I7-6850K, a GPU of NVIDIA RTX2070 and a 32G RAM.

Training setting

These three one-stage models and three two-stage models were trained using the same hyper-parameters and dataset. Except for the loss function and weights of each class, the Adam optimizer [62] is applied with a learning rate of 0.01. The learning rate will decay to one tenth of itself every 50 epochs, and the number of epochs is set to 100. The whole dataset of 113 images are randomly split into training set (93 images) and test set (20 images). Given this dataset is quite small, transfer learning was applied. Specifically, backend models pre-trained on Pascal VOC 2012, SBD and Cityscapes datasets [58] were used. The batch size in the training phase is six images considering the RAM of GPU. When fed into the model, all images are resized to a resolution of 900 × 900 for the one-stage models, and 200 × 200 for two-stage models. For images with different sizes, the *eps* of DBSCAN is automatically set to $\frac{\sqrt{\#Pixels}}{20}$ based on empirical observations. For data augmentation, the color parameters (brightness, saturation, hue and contrast) of the training images are randomly changed within a small range when training the model to avoid overfitting. As reflected by Fig. 7, all three models have successfully converged after 100 epochs in the training phase, during which losses of both Model1 and Model3 are smooth. Model2's loss is volatile and consistently higher than other two models. The relevant code can be found at [63].

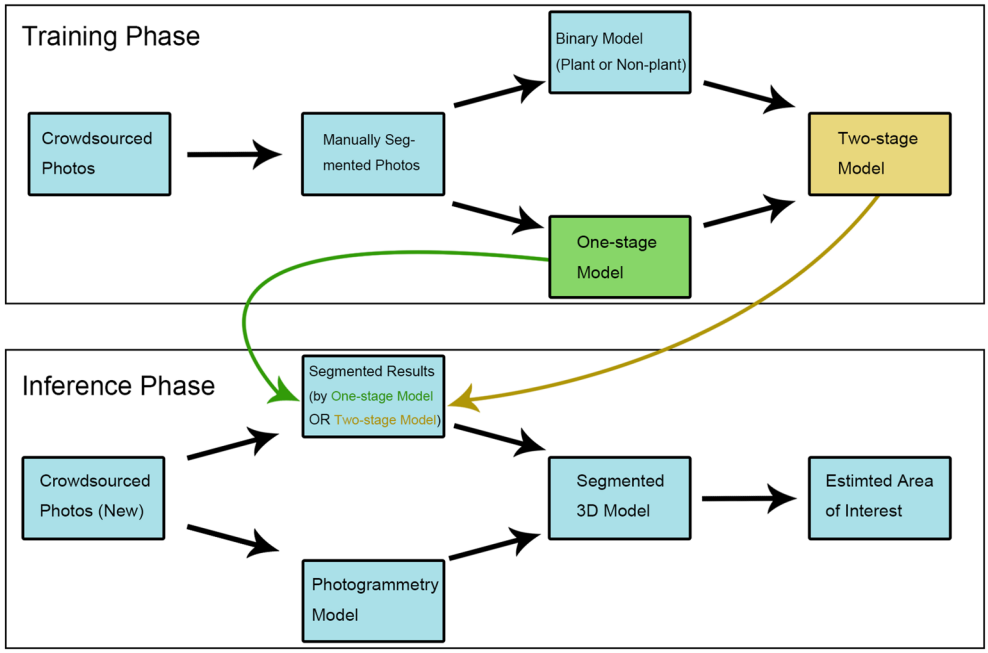

Fig. 6 A flowchart for the whole process of the proposed algorithm. At the training phase, crowdsourced photos are collected and manually segmented to train one-stage and two-stage models. In the inference phase, the newly collected crowdsourced photos will be processed by the (one-stage or two-stage) image segmentation model and a photogrammetry model to obtain a segmented 3D model, which can be used for further estimation of area of interest

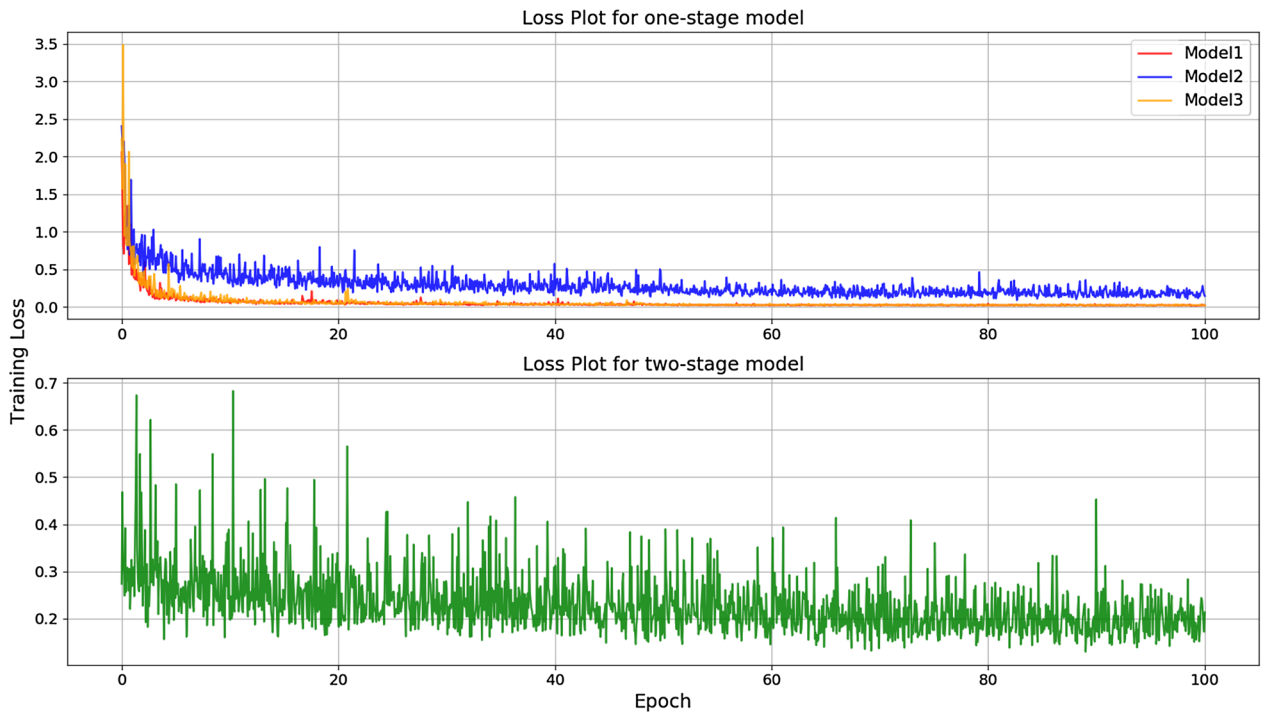

Fig. 7 The loss plot for three one-stage models and two-stage model in the training phase

Results and discussion

Results

$$Precision = \frac{1}{K}\sum_{j=1}^{K}\frac{TP_j}{TP_j + FP_j} \tag{3}$$

$$Recall = \frac{1}{K}\sum_{j=1}^{K}\frac{TP_j}{TP_j + FN_j} \tag{4}$$

$$IoU = \frac{1}{K}\sum_{j=1}^{K}\frac{TP_j}{TP_j + FN_j + FP_j} \tag{5}$$

$$MPA = \frac{1}{K}\sum_{j=1}^{K}\frac{n_{jj}}{t_j} \text{ for } K \text{ appear in the image} \tag{6}$$

In this article, five metrics are used to evaluate the performance of the model, namely Precision, Recall, Intersection over Union (IoU or Jaccard Index), F1-score and Mean Pixel Accuracy (MPA) [64]. As the first four metrics are only defined for binary classification case, the macro-average of these metrics for each class j is used. Specifically, in binary classification, if define TP_j, FP_j, FN_j and TN_j according to the confusion matrix in Table 1 for class j, Precision, Recall and IoU can be defined by Eqs. (3–5), where K is the number of classes. F1-score is the harmonic mean of Precision and Recall. MPA is the macro-averaged pixel accuracy for all classes, which is shown in Eq. 6, where n_{jj} is the total number of TPs for

Table 1 The definition of elements in confusion matrix for class j

	$Prediction_j$	
	Negative	**Positive**
$Label_j$		
Negative	True negative (TN_j)	False positive (FP_j)
Positive	False negative (FN_j)	True positive (TP_j)

$Prediction_j$ and $Label_j$ are predictions and true labels for class j

class j and t_j is the total number of pixels labelled as class j. These metrics avoid biases caused by unbalanced number of pixels between different classes.

Under the hardware condition described in "Software and hardware" section, the training process took approximately 1 h for the training set with a size of 93 images in this case. As for the inference phase, the majority of the time spent on prediction of all three trained models is less than 0.2 s in the first stage and 0.6 s for the two-stage task. This demonstrates the practicality of the models in terms of required processing time.

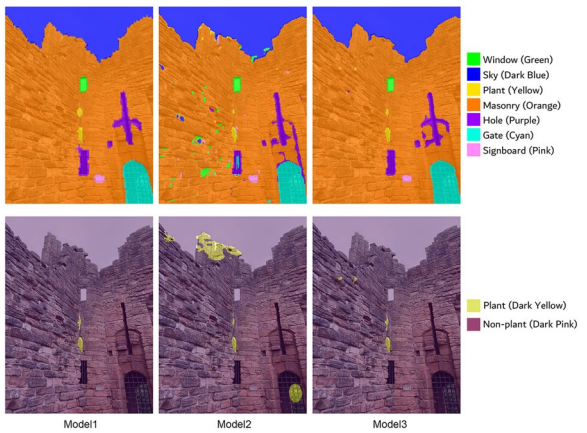

Fig. 8 The visual representation of the segmented photographs. The first row are the results of the one-stage model, and the second row are the results of two-stage model

Figure 8 shows the predictions of the one-stage (first row) and two-stage (second row) models overlapped with the original image for a typical example in the test set. As displayed in Fig. 8, the prediction of Model2 contains more false positives (FPs) compared to Model1 and Model3. The boxplot in Fig. 9 summarizes the performance of the three models on the test set with a confidence interval of 95% and a whisker of 1.5. All the metrics have been averaged across 20 test images. Note that metrics are not calculated for a class if there are no groundtruth labels for that class in the test image. In this case, there were two test images of the inner corner of Bothwell Castle that did not have plants in them.

Among the three models, Model3 has the best performance in all metrics over other models reflected by its overlall IoU of 66.9% (one-stage plant 54.6%, two-stage plant 56.2%), overall F1-score of 74.3% (one-stage plant 69.6%, two-stage plant 70.5%) as well as overall MPA of

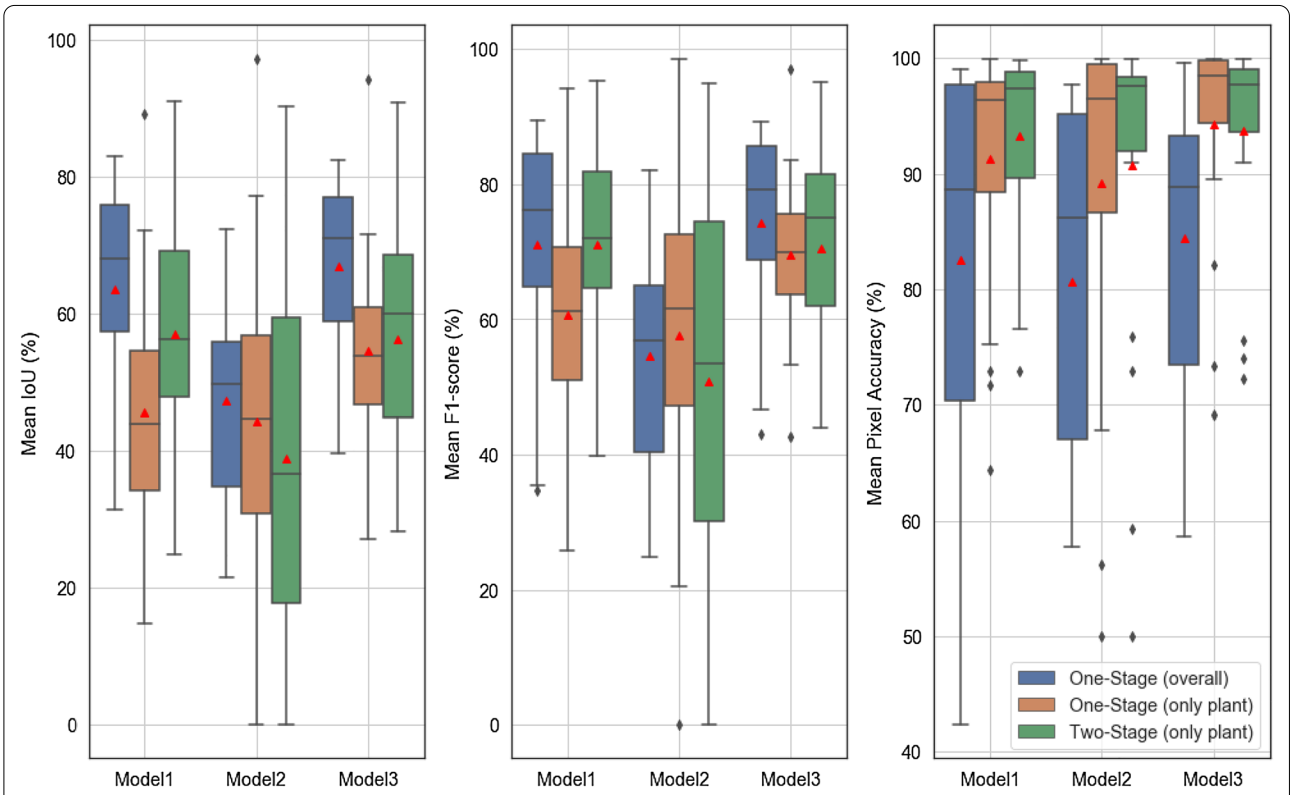

Fig. 9 The boxplot of the results of one-stage model and two-stage model's performance on overall classes and plant class, where red triangles represent mean values. IoU and F1-score are not accounted if there are no groundtruth labels for that class in the image

Table 2 Median of differences between different models

	IoU (%)	F1 (%)	MPA (%)	Precision (%)	Recall (%)
Overall					
M1–M2	14.6 (5.3, 25.6)	15.3 (5.0, 25.4)	1.1 (− 3.1, 7.0)	14.4 (4.6, 27.5)	13.9 (1.0, 17.6)
M1–M3	− 2.3 (− 5.1, 0.3)	− 2.3 (− 3.5, 0.0)	− 1.1 (− 3.9, 0.6)	− 2.7 (− 3.6, 0.3)	− 1.4 (− 4.2, 0.6)
M2–M3	− 21.3 (− 26.1, − 9.7)	− 20.7 (− 26.6, − 7.7)	− 2.0 (− 6.3, 1.3)	− 21.2 (− 29.1, − 9.0)	− 12.9 (− 19.2, − 2.5)
Plant (1s)					
M1–M2	− 0.9 (− 11.8, 8.5)	− 0.8 (− 10.7, 8.3)	0.7 (− 2.4, 2.9)	− 6.5 (− 13.6, 10.0)	2.3 (− 4.6, 6.0)
M1–M3	− 8.8 (− 14.6, − 1.0)	− 9.0 (− 16.0, − 0.8)	− 1.7 (− 3.8, − 0.2)	− 7.8 (− 14.8, 0.4)	− 3.3 (− 8.7, − 0.1)
M2–M3	− 6.2 (− 14.7, 3.0)	− 4.9 (− 16.2, 2.4)	− 0.7 (− 5.5, 0.2)	2.7 (− 12.5, 5.6)	− 2.4 (− 12.5, 0.1)
Plant (2s)					
M1–M2	10.9 (6.5, 25.2)	9.6 (6.7, 27.2)	0.1 (− 0.3, 1.1)	13.4 (7.4, 22.1)	0.2 (− 0.4, 1.9)
M1–M3	2.9 (− 3.7, 5.7)	2.2 (− 3.5, 4.7)	− 0.1 (− 1.4, 0.5)	1.5 (− 3.8, 5.5)	− 0.2 (− 3.1, 1.1)
M2–M3	− 10.5 (− 26.1, − 2.6)	− 10.2 (− 29.7, − 2.6)	− 0.3 (− 1.0, 0.1)	− 10.4 (− 23.1, − 2.9)	− 0.4 (− 1.9, − 0.0)

The numbers in brackets are the 25th and 75th percentiles

84.4% (one-stage plant 94.3%, two-stage plant 93.7%). Although Model1's performance and Model3's performance are close as reflected by Table 2, Model3's prediction is smoother and able to detect small objects such as the plants at the left-upper corner in Fig. 8. Therefore, Model3 is chosen as the default model for this task.

Figure 10 shows the confusion matrices for Model3, including precision rates and recall rates of each class, for one-stage and two-stage models on the left and right respectively. For the plant class, the precision rate is consistently lower than the recall rate. This indicates that the models tend to aggressively predict pixels to the plant class, resulting in a large number of FPs to lower the precision rate. By comparing the left matrix and right matrix in Fig. 10, and considering the values in Table 3, the second stage in the model is shown to slightly improve the precision rate but decrease the recall rate for Model3. Figure 11 presents the generalisability of Model3 to segmenting plant classes from images of other scenes besides the Bothwell castle (with *eps* for DBSCAN set to 3).

Finally, Fig. 12 displays the reconstructed 3D photogrammetry models. Specifically, 87 out of 93 images in the training set are used by VisualSFM to build the 3D photogrammetry model. In Fig. 12, from left to right is the original 3D photogrammetry model, the 3D photogrammetry model textured with overall classes, and the 3D photogrammetry model textured with a binary mask for the plant class. As presented in Fig. 12, the final 3D model with segmented textures clearly gives the partitions of each object in this scene with fairly high accuracy in 3D space, which allows the properties, such as area, of the interested object to be accurately and easily monitored in arbitrary perspective. The percentage below the name of each class in Fig. 12 is the ratio of the estimated count of pixels over that of the whole model surface. Given these ratios, the area of interest can be easily calculated when the area of a reference object is known.

Discussion

From the results presented above, several advantages of the model could be summarized as follows. Firstly, the accuracy of the segmented results for an unstructured and complex scenario is reasonable, demonstrated by the performance statistics and the visualization of the output result. Secondly, the average processing time per image could be considered acceptable even for larger projects. Thirdly, training the model is relatively straightforward and object features are automatically extracted by CNN layers, instead of a cumbersome manual feature extraction processes. Lastly, with the assistance of

Table 3 Median of differences between one-stage models and two-stage models (1s and 2s represent one-stage model and two-stage model respectively)

	IoU (%)	F1 (%)	MPA (%)	Precision (%)	Recall (%)
2s–1s (M1)	12.2 (2.0, 28.1)	10.7 (1.2, 22.9)	0.3 (− 1.5, 2.4)	14.7 (1.9, 28.6)	0.4 (− 4.4, 5.2)
2s–1s (M2)	− 3.5 (− 10.6, 1.6)	− 1.9 (− 9.8, 1.7)	0.0 (− 0.8, 1.6)	− 5.0 (− 14.3, 1.1)	0.8 (− 1.2, 5.3)
2s–1s (M3)	4.9 (− 3.0, 9.2)	3.6 (− 1.7, 7.9)	− 0.6 (− 1.9, 0.9)	7.9 (− 2.5, 12.1)	− 1.3 (− 4.3, 3.4)

The numbers in brackets are the 25th and 75th percentiles

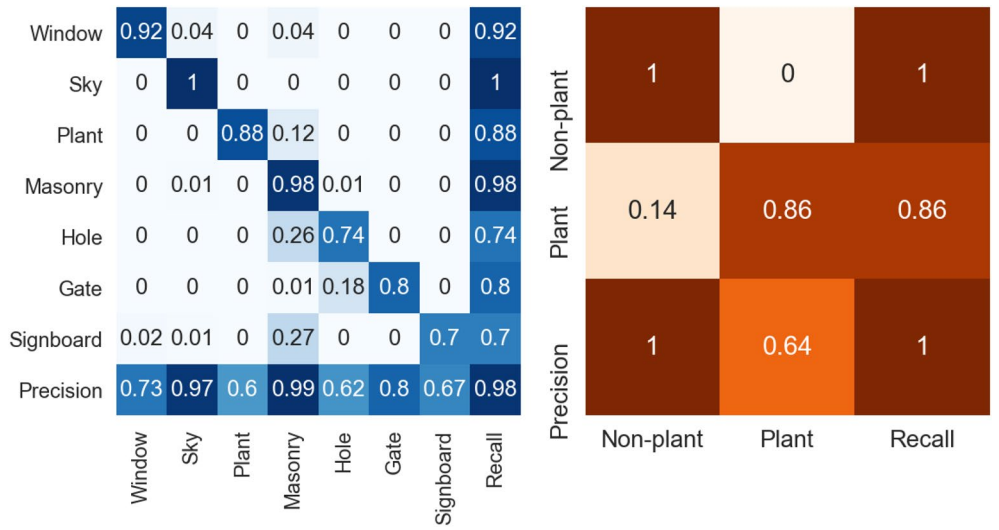

Fig. 10 Confusion matrix of Model3 for one-stage model and two-stage model. All data are averaged across 20 test images, and are normalized by column. Precision and Recall are not accounted if no groundtruth labels for that class in the image

photogrammetry technique, the segmented images could be synthesized into a 3D model with segmented textures, which allows the properties such as area, shape or volume to be measured from different perspectives, eliminating the inaccuracy and biases introduced by different angles of shooting.

However, there are three main limitations associated with this proposed model. The first one is the difficulty in small objects detection. This is inherited from all deep-learning based computer vision algorithms that have to make trade-offs between global and local vision [50]. More specifically, since the model will process the input image into a feature map, which is smaller than the original image by a factor of the output stride defined by the model in terms of the resolution, the model will fail to detect or give an imprecise result to very small objects. The second disadvantage of the model is the large number of FPs in the predictions, which may lead to an overestimation of the plant area. The last limitation for the model is the requirement of photographs with comprehensive angles taken within a specific period of time. If the interval of time between photographs used to build the photogrammetry model is too long, the built 3D model of the scene may not be representative for any particular timestamp.

Further work and potential applications

There are potential future directions for the model to be improved. The first way to strengthen the performance of the model is to increase the size of the training examples used to fine-tune the classifier part in the model. It should be noted that the current image segmentation model has been only trained by 93 training images, which is an extremely small size of training dataset. A larger size of training dataset would definitely benefit the model considering the complicated model structure. The additional training images for the image segmentation model should not be limited to photographs of this site. Images from other scenes could be also beneficial to the model as long as they contain similar objects due to the advanced generalization ability of deep-learning based models as shown by Fig. 11. Secondly, in this paper, the last photogrammetry part may be further refined by supplementing other information. For example, some photographs used to build the photogrammetry model lack EXIF data. Thus, real focal length data of those photographs is missing. Although the SfM algorithm will automatically set focal length to a medium viewing angle for photographs missing EXIF data [60], more accurate focal length data could help to ensure the quality and consistency of the result from photogrammetry algorithm [65].

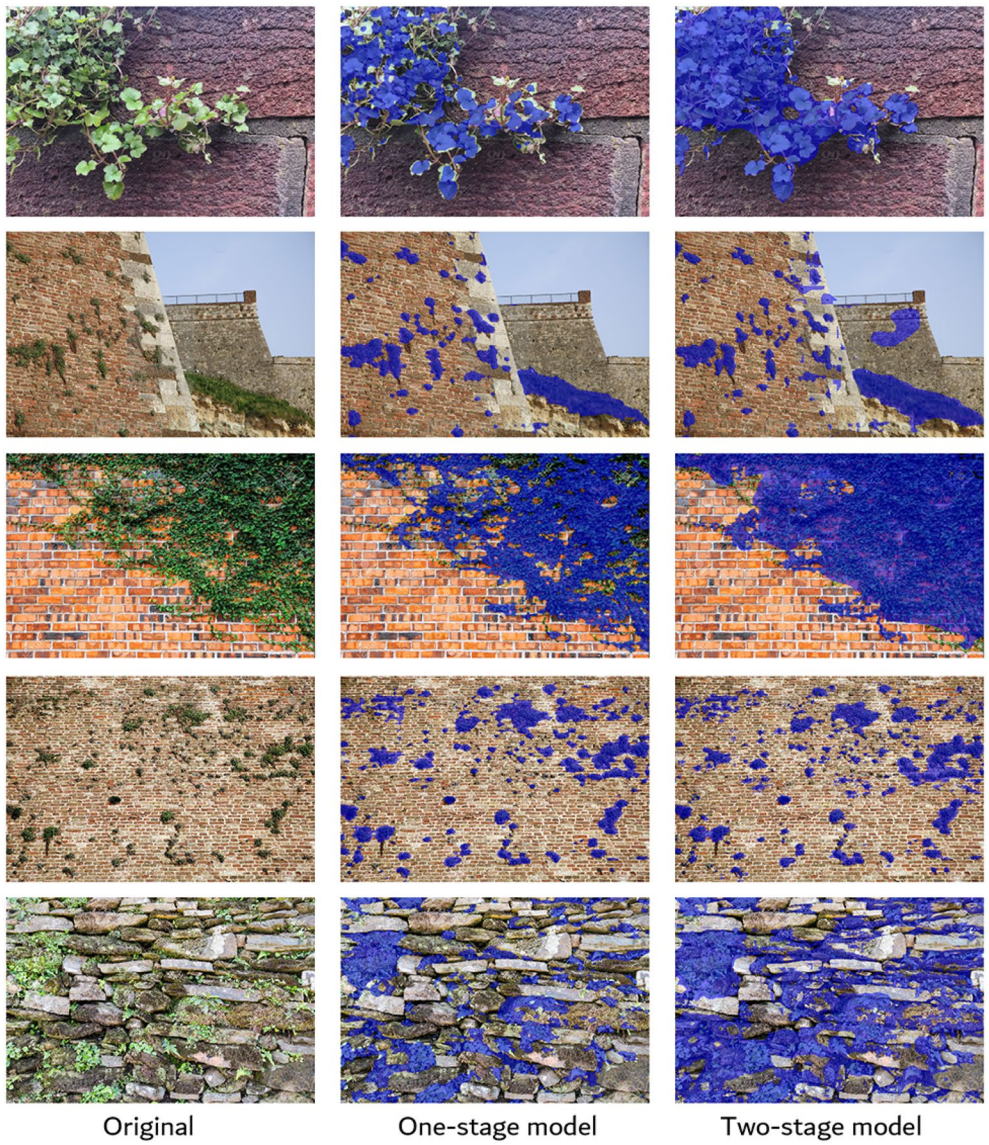

Fig. 11 Segmentation results for plant class in other scenes (predicted regions are highlighted in blue). The *eps* for DBSCAN is set to 3

In terms of its other potential applications, this image segmentation model can be easily changed to monitor other damages in historical sites, such as cracks, erosion and insects based on 2D image data. This transformation can be done by re-training the model on the new training images and their correspondent labels. Beyond this, the model could potentially be used as a semantic masking tool in photogrammetry workflows and applications [66]. This is achieved by using image segmentation model to automatically partition the object of interest from the background in images before processing them with photogrammetry algorithms.

Conclusion

To extract useful information from the increasing amount of crowdsourced cultural heritage photographs uploaded to online websites by visitors and amateurs, this article proposes to use DeeplabV3+, a deep-learning based semantic image segmentation

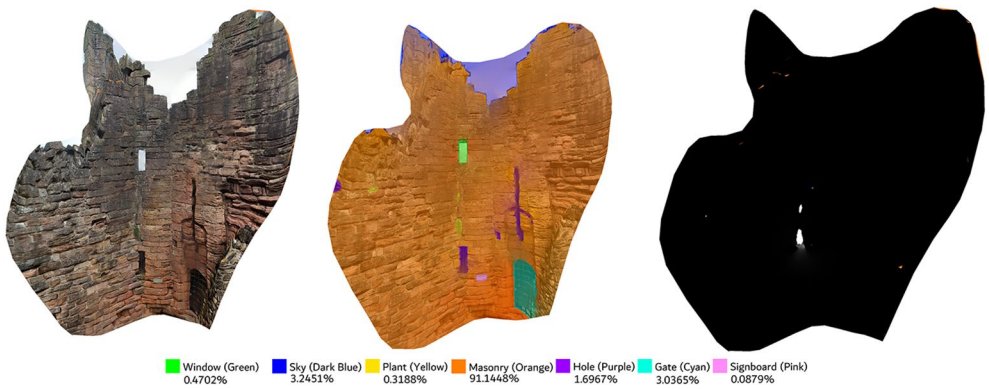

Fig. 12 The front view of the built segmented 3D photogrammetry model. They are the original model, the model with segmented texture for overall classes and the model with texture for plant class from left to right. The percentage below the name of each class is the ratio of the estimated count of pixels over that of the whole model surface

algorithm that is capable of handling segmentation tasks in complex scenarios. It is further combined with photogrammetry to automatically process the crowdsourced photographs. This can significantly save human and capital resources spent on on-site inspection and management of remote built heritage sites.

Although there are still some limitations such as the difficulty in segmenting small objects and relatively low precision rate, on the task of monitoring the plant growth in Bothwell Castle, this model has a satisfying segmentation performance in terms of the metrics of an overall IoU of 66.9% (one-stage plant 54.6%, two-stage plant 56.2%), an overall F1-score of 74.3% (one-stage plant 69.6%, two-stage plant 70.5%) as well as an overall MPA of 84.4% (one-stage plant 94.3%, two-stage plant 93.7%). Besides, this model has a powerful generalisability in other scenes besides Bothwell castle and an acceptable processing time per image of less than 0.2 s for one-stage model and 0.6 s for two-stage model on a home PC. This demonstrates the usability and practicality of its application in the cultural heritage sector. In addition, the segmented results are successfully combined with built photogrammetry models to measure the area of the plant from arbitrary perspectives, eliminating distortions caused by angle of shooting.

This method can be easily transferred to other monitoring tasks to measure cracks, erosion or even insects by replacing the training images and adjusting the classes in the model. Beyond area measurement, other properties of these objects, such as volume and shape, can be potentially measured by using image segmentation as an automated masking tool in the pre-processing step of photogrammetry. In conclusion, the proposed combination of semantic segmentation and photogrammetry can be effectively and efficiently applied to automatically extract useful information from crowdsourced data to support remote cultural heritage sites monitoring activities.

Appendix A: Model architecture

Model architecture in terms of convolutional layers and pooling layers for each model used in this research is shown in Table 4, where square brackets represent processing blocks.

Table 4 Model architecture in terms of convolutional layers and pooling layers for each model used in this research

	Model1	Model2	Model3	Two-stage model
Backbone	$\left[\left[(7\times 7;\, s=2;\, r=1),\, 64\right]\times 1\right]$	$\left[\left[(3\times 3;\, s=2;\, r=1),\, 32\right]\times 1\right]$	$\left[\left[(7\times 7;\, s=2;\, r=1),\, 64\right]\times 1\right]$	$\left[\left[(7\times 7;\, s=2;\, r=1),\, 64\right]\times 1\right]$
	$\left[MaxPool2d(3\times 3;\, s=2;\, r=1)\right]$	$\left[\left[(3\times 3;\, s=1;\, r=1),\, 64\right]\times 1\right]$	$\left[MaxPool2d(3\times 3;\, s=2;\, r=1)\right]$	$\left[MaxPool2d(3\times 3;\, s=2;\, r=1)\right]$
	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 64\right]\times 3\\ \left[(3\times 3;\, s=1;\, r=1),\, 64\right]\times 3\\ \left[(1\times 1;\, s=1;\, r=1),\, 256\right]\times 4\end{bmatrix}$	$\left[\left[(1\times 1;\, s=2;\, r=1),\, 128\right]\times 1\right]$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 64\right]\times 3\\ \left[(3\times 3;\, s=1;\, r=1),\, 64\right]\times 3\\ \left[(1\times 1;\, s=1;\, r=1),\, 256\right]\times 4\end{bmatrix}$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 64\right]\times 3\\ \left[(3\times 3;\, s=1;\, r=1),\, 64\right]\times 3\\ \left[(1\times 1;\, s=1;\, r=1),\, 256\right]\times 4\end{bmatrix}$
	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 128\right]\times 4\\ \left[(3\times 3;\, s=2;\, r=1),\, 128\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 512\right]\times 4\\ \left[(1\times 1;\, s=2;\, r=1),\, 512\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=1),\, 128\right]\times 3\end{bmatrix}$	$\begin{bmatrix}\begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 64\\ (1\times 1;\, s=1;\, r=1),\, 128\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 128\\ (1\times 1;\, s=1;\, r=1),\, 128\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=2;\, r=1),\, 128\\ (1\times 1;\, s=1;\, r=1),\, 128\end{bmatrix}\times 1\end{bmatrix}$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 128\right]\times 4\\ \left[(3\times 3;\, s=2;\, r=1),\, 128\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 512\right]\times 4\\ \left[(1\times 1;\, s=2;\, r=1),\, 512\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=1),\, 128\right]\times 3\end{bmatrix}$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 128\right]\times 4\\ \left[(3\times 3;\, s=2;\, r=1),\, 128\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 512\right]\times 4\\ \left[(1\times 1;\, s=2;\, r=1),\, 512\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=1),\, 128\right]\times 3\end{bmatrix}$
	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 256\right]\times 23\\ \left[(3\times 3;\, s=1;\, r=2),\, 256\right]\times 23\\ \left[(1\times 1;\, s=1;\, r=1),\, 1024\right]\times 24\end{bmatrix}$	$\left[\left[(1\times 1;\, s=2;\, r=1),\, 256\right]\times 1\right]$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 256\right]\times 23\\ \left[(3\times 3;\, s=1;\, r=2),\, 256\right]\times 23\\ \left[(1\times 1;\, s=1;\, r=1),\, 1024\right]\times 24\end{bmatrix}$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 256\right]\times 23\\ \left[(3\times 3;\, s=2;\, r=1),\, 256\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 1024\right]\times 23\\ \left[(1\times 1;\, s=2;\, r=1),\, 1024\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=1),\, 256\right]\times 22\end{bmatrix}$
	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 512\right]\times 3\\ \left[(3\times 3;\, s=1;\, r=4),\, 512\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 2048\right]\times 4\\ \left[(3\times 3;\, s=1;\, r=8),\, 512\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=16),\, 512\right]\times 1\end{bmatrix}$	$\begin{bmatrix}\begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 128\\ (1\times 1;\, s=1;\, r=1),\, 256\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 256\\ (1\times 1;\, s=1;\, r=1),\, 256\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=2;\, r=1),\, 256\\ (1\times 1;\, s=1;\, r=1),\, 256\end{bmatrix}\times 1\end{bmatrix}$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 512\right]\times 3\\ \left[(3\times 3;\, s=1;\, r=4),\, 512\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 2048\right]\times 4\\ \left[(3\times 3;\, s=1;\, r=8),\, 512\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=16),\, 512\right]\times 1\end{bmatrix}$	$\begin{bmatrix}\left[(1\times 1;\, s=1;\, r=1),\, 512\right]\times 3\\ \left[(3\times 3;\, s=1;\, r=2),\, 512\right]\times 1\\ \left[(1\times 1;\, s=1;\, r=1),\, 2048\right]\times 4\\ \left[(3\times 3;\, s=1;\, r=4),\, 512\right]\times 1\\ \left[(3\times 3;\, s=1;\, r=8),\, 512\right]\times 1\end{bmatrix}$
		$\left[\left[(1\times 1;\, s=2;\, r=1),\, 728\right]\times 1\right]$		
		$\begin{bmatrix}\begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 256\\ (1\times 1;\, s=1;\, r=1),\, 728\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 728\\ (1\times 1;\, s=1;\, r=1),\, 728\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=2;\, r=1),\, 728\\ (1\times 1;\, s=1;\, r=1),\, 728\end{bmatrix}\times 1\end{bmatrix}$		
		$\begin{bmatrix}\left[(3\times 3;\, s=1;\, r=1),\, 728\right]\\ \left[(1\times 1;\, s=1;\, r=1),\, 728\right]\end{bmatrix}\times 48$		
		$\left[\left[(1\times 1;\, s=1;\, r=1),\, 1024\right]\times 1\right]$		
		$\begin{bmatrix}\begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 728\\ (1\times 1;\, s=1;\, r=1),\, 728\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 728\\ (1\times 1;\, s=1;\, r=1),\, 1024\end{bmatrix}\times 1\\ \begin{bmatrix}(3\times 3;\, s=1;\, r=1),\, 1024\\ (1\times 1;\, s=1;\, r=1),\, 1024\end{bmatrix}\times 1\end{bmatrix}$		
		$\left[\left[(3\times 3;\, s=1;\, r=2),\, 1024\right]\times 1\right]$		

Table 4 (continued)

	Model1	Model2	Model3	Two-stage model
		$[[(1\times1; s=1; r=1), 1536]\times1]$		
		$[[(3\times3; s=1; r=2), 1536]\times1]$		
		$[[(1\times1; s=1; r=1), 1536]\times1]$		
		$[[(3\times3; s=1; r=2), 1536]\times1]$		
		$[[(1\times1; s=1; r=1), 2048]\times1]$		
Classifier	$[[(1\times1; s=1; r=1), 256]\times1]$	$[[(1\times1; s=1; r=1), 256]\times1]$	$[[(1\times1; s=1; r=1), 256]\times1]$	$[[(1\times1; s=1; r=1), 256]\times1]$
	$[[(3\times3; s=1; r=12), 256]\times1]$	$[[(3\times3; s=1; r=6), 256]\times1]$	$[[(3\times3; s=1; r=12), 256]\times1]$	$[[(3\times3; s=1; r=6), 256]\times1]$
	$[[(3\times3; s=1; r=24), 256]\times1]$	$[[(3\times3; s=1; r=12), 256]\times1]$	$[[(3\times3; s=1; r=24), 256]\times1]$	$[[(3\times3; s=1; r=12), 256]\times1]$
	$\begin{bmatrix}[(3\times3; s=1; r=36), 256]\times1\\ [(1\times1; s=1; r=1), 256]\times1\end{bmatrix}$	$[[(3\times3; s=1; r=18), 256]\times1]$	$[[(3\times3; s=1; r=36), 256]\times1]$	$[[(3\times3; s=1; r=18), 256]\times1]$
	$\begin{bmatrix}[AdaptiveAvgPool2d(1\times1)]\\ [(1\times1; s=1; r=1), 256]\times1\end{bmatrix}$	$\begin{bmatrix}[AdaptiveAvgPool2d(1\times1)]\\ [(1\times1; s=1; r=1), 256]\times1\end{bmatrix}$	$\begin{bmatrix}[AdaptiveAvgPool2d(1\times1)]\\ [(1\times1; s=1; r=1), 256]\times1\end{bmatrix}$	$\begin{bmatrix}[AdaptiveAvgPool2d(1\times1)]\\ [(1\times1; s=1; r=1), 256]\times1\end{bmatrix}$
	$[[(3\times3; s=1; r=1), 256]\times1]$	$[[(1\times1; s=1; r=1), 256]\times1]$	$[[(1\times1; s=1; r=1), 256]\times1]$	$[[(1\times1; s=1; r=1), 256]\times1]$
	$[[(1\times1; s=1; r=1), 8]\times1]$	$[[(1\times1; s=1; r=1), 48]\times1]$	$[[(1\times1; s=1; r=1), 48]\times1]$	$[[(1\times1; s=1; r=1), 48]\times1]$
		$[[(3\times3; s=1; r=1), 256]\times1]$	$[[(3\times3; s=1; r=1), 256]\times1]$	$[[(3\times3; s=1; r=1), 256]\times1]$
		$[[(3\times3; s=1; r=1), 256]\times1]$	$[[(3\times3; s=1; r=1), 256]\times1]$	$[[(3\times3; s=1; r=1), 256]\times1]$
		$[[(1\times1; s=1; r=1), 8]\times1]$	$[[(1\times1; s=1; r=1), 8]\times1]$	$[[(1\times1; s=1; r=1), 2]\times1]$

Square brackets represent processing blocks. The format for each processing block is [(*kernel size*; *stride*; *rate of dilation*), *output dimension*]

Acknowledgements
The authors would address their thanks to all anonymous contributors who uploaded crowdsourced photographs used in this research.

Authors' contributions
ZL is responsible for the design of the image segmentation model and segmentation-photogrammetry workflow. RB supervised the work and provided the data. ERL investigated the research background and summarized past research. LW and AF collected and managed the data. SAO and JG supervised the work and revised the manuscript. All authors read and approved the fnal manuscript.

Funding
This research has been funded by Engineering and Physical Sciences Research Council (Reference number: EP/L016036/1) under the name of Miss Rosie Brigham

Data availability
Please contact the corresponding author for data requests.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]University College London, Central House, 14 Upper Woburn Place, London WC1H 0NN, UK. [2]Historic Environment Scotland, Salisbury Place, Longmore House, Edinburgh EH9 1SH, UK.

Received: 25 October 2021 Accepted: 5 February 2022
Published online: 19 February 2022

References

1. Historic Environment Scotland. Historic Environment Scotland—PLACES TO VISIT. 2020. https://members.historic-scotland.gov.uk/places. Accessed 1 Mar 2021.
2. Historic Environment Scotland. About historic environment Scotland. 2020. https://www.historicenvironment.scot/about-us/who-we-are/about-historic-environment-scotland/. Accessed 1 Mar 2021.
3. Mesquita E, Antunes P, Coelho F, André P, Arêde A, Varum H. Global overview on advances in structural health monitoring platforms. J Civil Struct Health Monit. 2016;6(3):461–75.
4. Mishra M. Machine learning techniques for structural health monitoring of heritage buildings: a state-of-the-art review and case studies. J Cult Herit. 2021;47:227–45.
5. Kumar P. Crowdsourcing to rescue cultural heritage during disasters: a case study of the 1966 florence flood. Int J Disaster Risk Reduct. 2020;43:101371.
6. Wilson AS, Gaffney V, Gaffney C, Ch'ng E, Bates R, Sears G, Sparrow T, Murgatroyd A, Faber E, Coningham RAE. Curious travellers: repurposing imagery to manage and interpret threatened monuments, sites and landscapes. In: Heritage under pressure—threats and solution: studies of agency and soft power in the historic environment. Oxbow Books; 2019.
7. Vincent ML. Crowdsourced data for cultural heritage. In: Vincent ML, Ioannides M, Levy TE, editors. Heritage and archaeology in the digital age. Berlin: Springer; 2017.
8. Barrington L, Ghosh S, Greene M, Har-Noy S, Berger J, Gill S, Lin AY-M, Huyck C. Crowdsourcing earthquake damage assessment using remote sensing imagery. Ann Geophys. 2011;54(6).
9. Zhou B, Lapedriza A, Xiao J, Torralba A, Oliva A. Learning deep features for scene recognition using places database. In: Advances in neural information processing systems; 2014. p. 487–95.
10. Wang N, Zhao Q, Li S, Zhao X, Zhao P. Damage classification for masonry historic structures using convolutional neural networks based on still images: damage classification for masonry historic structures using cnns. Comput-Aided Civil Infrastruct Eng. 2018;33(12):1073–89. https://doi.org/10.1111/mice.12411.
11. Wang N, Zhao X, Zhao P, Zhang Y, Zou Z, Ou J. Automatic damage detection of historic masonry buildings based on mobile deep learning. Autom Constr. 2019;103:53–66. https://doi.org/10.1016/j.autcon.2019.03.003.
12. Sakamoto M, Tsuguchi M, Chhatkuli S, Satoh T. Extended multiscale image segmentation for castellated wall management. Int Arch Photogramm Remote Sens Spat Inf Sci ISPRS Arch. 2018;42(2):999–1005.
13. Sakamoto M, Shinohara T, Li Y, Satoh T. Wall stone extraction based on stacked conditional gan and multiscale image segmentation. Int Arch Photogramm Remote Sens Spat Inf Sci. 2020;XLIII–B2–2020:1491–6. https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-1491-2020.
14. Ibrahim Y, Nagy B, Benedek C. Deep learning-based masonry wall image analysis. Remote Sens (Basel, Switzerland). 2020;12(3918):3918. https://doi.org/10.3390/rs12233918.
15. Perez H, Tah JHM, Mosavi A. Deep learning for detecting building defects using convolutional neural networks. Sensors (Basel, Switzerland). 2019;19(16):3556. https://doi.org/10.20944/preprints201908.0068.v1.
16. Pezzica C, Schroeter J, Prizeman OE, Jones CB, Rosin PL. Between images and built form: automating the recognition of standardised building components using deep learning. ISPRS Ann Photogramm Remote Sens Spat Inf Sci. 2019;4(2):123–32. https://doi.org/10.5194/isprs-annals-IV-2-W6-123-2019.
17. Zou Z, Zhao X, Zhao P, Qi F, Wang N. CNN-based statistics and location estimation of missing components in routine inspection of historic buildings. J Cult Herit. 2019;38:221–30. https://doi.org/10.1016/j.culher.2019.02.002.
18. Wang N, Zhao X, Zou Z, Zhao P, Qi F. Autonomous damage segmentation and measurement of glazed tiles in historic buildings via deep learning. Comput-Aided Civil Infrastruct Eng. 2020;35(3):277–91. https://doi.org/10.1111/mice.12488.
19. Hatir ME, Barstuğan M, İnce İ. Deep learning-based weathering type recognition in historical stone monuments. J Cult Herit. 2020. https://doi.org/10.1016/j.culher.2020.04.008.
20. Easlon HM, Bloom AJ. Easy leaf area: automated digital image analysis for rapid and accurate measurement of leaf area. Appl Plant Sci. 2014;2(7):1400033. https://doi.org/10.3732/apps.1400033.
21. Schrader J, Pillar G, Kreft H. Leaf-it: an android application for measuring leaf area. Ecol Evol. 2017;7(22):9731–8. https://doi.org/10.1002/ece3.3485.
22. Cerimele MM, Cossu R. A numerical modelling for the extraction of decay regions from color images of monuments. Math Comput Simul. 2009;79(8):2334–44. https://doi.org/10.1016/j.matcom.2009.01.015.
23. Manferdini AM, Baroncini V, Corsi C. An integrated and automated segmentation approach to deteriorated regions recognition on 3d reality-based models of cultural heritage artifacts. J Cult Herit. 2012;13(4):371–8. https://doi.org/10.5194/10.1016/j.culher.2012.01.014.
24. Chen Z, Ting D, Newbury R, Chen C. Semantic segmentation for partially occluded apple trees based on deep learning. Comput Electron Agric. 2021;181:105952. https://doi.org/10.1016/j.compag.2020.105952.
25. Voulodimos A, Doulamis N, Fritsch D, Makantasis K, Doulamis A, Klein M. Four-dimensional reconstruction of cultural heritage sites based on photogrammetry and clustering. J Electron Imaging. 2016;26(1):011013. https://doi.org/10.1117/1.JEI.26.1.011013.
26. Wilson L, Rawlinson A, Frost A, Hepher J. 3d digital documentation for disaster management in historic buildings: applications following fire damage at the mackintosh building, the glasgow school of art. J Cult Herit. 2018;31:24–32. https://doi.org/10.1016/j.culher.2017.11.012.
27. Yang X, Grussenmeyer P, Koehl M, Macher H, Murtiyoso A, Landes T. Review of built heritage modelling: integration of hbim and other information techniques. J Cult Herit. 2020;46:350–60.
28. Valero E, Forster A, Bosché F, Hyslop E, Wilson L, Turmel A. Automated defect detection and classification in ashlar masonry walls using machine learning. Autom Constr. 2019;106:102846. https://doi.org/10.1016/j.autcon.2019.102846.
29. Randazzo L, Collina M, Ricca M, Barbieri L, Bruno F, Arcudi A, La Russa MF. Damage indices and photogrammetry for decay assessment of stone-built cultural heritage: the case study of the san domenico church main entrance portal (south calabria, italy). Sustainability (Basel, Switzerland). 2020;12(12):5198. https://doi.org/10.3390/su12125198.
30. Barsanti SG, Guidi G, De Luca L. Segmentation of 3d models for cultural heritage structural analysis-some critical issues. ISPRS Ann Photogramm

Remote Sens Spat Inf Sci. 2017;4:115. https://doi.org/10.5194/isprs-annals-IV-2-W2-115-2017.
31. Abate D. Built-heritage multi-temporal monitoring through photogrammetry and 2d/3d change detection algorithms. Stud Conserv. 2019;64(7):423–34. https://doi.org/10.1080/00393630.2018.1554934.
32. Morbidoni C, Pierdicca R, Quattrini R, Frontoni E. Graph cnn with radius distance for semantic segmentation of historical buildings tls point clouds. Int Arch Photogramm Remote Sens Spat Inf Sci. 2020;XLIV–4–W1–2020:95–102.
33. Teruggi S, Grilli E, Russo M, Fassi F, Remondino F. A hierarchical machine learning approach for multi-level and multi-resolution 3d point cloud classification. Remote Sens (Basel, Switzerland). 2020;12(16):2598. https://doi.org/10.3390/rs12162598.
34. Grilli E, Remondino F. Machine learning generalisation across different 3d architectural heritage. ISPRS Int J Geo-inf. 2020;9(6):379. https://doi.org/10.3390/ijgi9060379.
35. Nousias S, Arvanitis G, Lalos AS, Pavlidis G, Koulamas C, Kalogeras A, Moustakas K. A saliency aware cnn-based 3d model simplification and compression framework for remote inspection of heritage sites. IEEE Access. 2020;8:169982–70001. https://doi.org/10.1109/ACCESS.2020.3023167.
36. Murtiyoso A, Grussenmeyer P. Virtual disassembling of historical edifices: experiments and assessments of an automatic approach for classifying multi-scalar point clouds into architectural elements. Sensors (Basel, Switzerland). 2020;20(8):2161. https://doi.org/10.3390/s20082161.
37. Monument Monitor. Monument monitor. 2020. https://www.monumentmonitor.co.uk/. Accessed 1 Mar 2021.
38. Historic Environment Scotland. Bothwell castle. 2020. https://www.historicenvironment.scot/visit-a-place/places/bothwell-castle/. Accessed 1 Mar 2021.
39. Labelbox. "*Labelbox*". 2020. https://labelbox.com/. Accessed 1 Mar 2021.
40. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems; 2012. p. 1097–105.
41. Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M. Deep learning for generic object detection: a survey. Int J Comput Vis. 2020;128(2):261–318. https://doi.org/10.1007/s11263-019-01247-4.
42. Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: a survey. 2020. arXiv preprint arXiv:2001.05566.
43. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick, CL. Microsoft coco: common objects in context. In: European conference on computer vision, Springer. 2014. p. 740–55. https://doi.org/10.1007/978-3-319-10602-1_48.
44. . Kuznetsova A, Rom H, Alldrin N, Uijlings J, Krasin I, Pont-Tuset J, Kamali S, Popov S, Malloci M, Duerig T, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. 2018. arXiv preprint arXiv:1811.00982. https://doi.org/10.1007/s11263-020-01316-z.
45. Fiorucci M, Khoroshiltseva M, Pontil M, Traviglia A, Del Bue A, James S. Machine learning for cultural heritage: a survey. Pattern Recogn Lett. 2020;133:102–8.
46. Chen L-C, Papandreou G, Schroff F, Adam H. Rethinking atrous convolution for semantic image segmentation. 2017. arXiv preprint arXiv:1706.05587.
47. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. p. 3431–40. https://doi.org/10.1109/CVPR.2015.7298965.
48. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention, Springer. 2015. p. 234–41. https://doi.org/10.1007/978-3-319-24574-4_28.
49. He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. 2017. p. 2961–9 . https://doi.org/10.1109/ICCV.2017.322.
50. Chen L-C, Papandreou G, Kokkinos I, Murphy K, Yuille AL. Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell. 2017;40(4):834–48. https://doi.org/10.1109/TPAMI.2017.2699184.
51. Chen L-C, Zhu Y, Papandreou G, Schroff F, Adam H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). 2018. p. 801–18.
52. Chollet F. Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. p. 1251–8. https://doi.org/10.1109/CVPR.2017.195.
53. Ester M, Kriegel H-P, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol. 96. 1996. p. 226–31.
54. Wu C. Towards linear-time incremental structure from motion. In: 2013 international conference on 3D vision-3DV 2013, IEEE. 2013. p. 127–34. https://doi.org/10.1109/3DV.2013.25.
55. Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vis. 2004;60(2):91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.
56. Triggs B, McLauchlan PF, Hartley RI, Fitzgibbon AW. Bundle adjustment-a modern synthesis. In: International workshop on vision algorithms, Springer. 1999. p. 298–372. https://doi.org/10.1007/3-540-44480-7_21.
57. Barnes C, Shechtman E, Finkelstein A, Goldman DB. PatchMatch: a randomized correspondence algorithm for structural image editing. 2009. https://doi.org/10.1145/1531326.1531330.
58. Zhang J. Pytorch-deeplab-xception. GitHub. 2019. https://github.com/jfzhang95/pytorch-deeplab-xception.
59. Bradski G. The OpenCV Library. Dr Dobb's J Software Tools. 2000.
60. Wu C, et al. VisualSFM: A visual structure from motion system. 2011.
61. Cernea D. OpenMVS: multi-view stereo reconstruction library. 2020. https://cdcseacave.github.io/openMVS.
62. Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. arXiv preprint arXiv:1412.6980.
63. Liu Z. Plant-segmentation. GitHub. 2021. https://github.com/sdyy6211/plant-segmentation. Accessed 1 Mar 2021.
64. Ulku I, Akagunduz E. A survey on deep learning-based architectures for semantic segmentation on 2d images. 2019. arXiv preprint arXiv:1912.10230.
65. Luhmann T. Precision potential of photogrammetric 6dof pose estimation with a single camera. ISPRS J Photogramm Remote Sens. 2009;64(3):275–84. https://doi.org/10.1016/j.isprsjprs.2009.01.002.
66. Stathopoulou E, Remondino F. Semantic photogrammetry: boosting image-based 3d reconstruction with semantic labeling. Int Arch Photogramm Remote Sens Spat Inf Sci. 2019;42(2):9. https://doi.org/10.5194/isprs-archives-XLII-2-W9-685-2019.

Publisher's Note